M 5

```
MMM         MMM     AAAAAAAAA        CCCCCCCCCCCC  RRRRRRRRRRRR        000000000
MMM         MMM     AAAAAAAAA        CCCCCCCCCCCC  RRRRRRRRRRRR        000000000
MMM         MMM     AAAAAAAAA        CCCCCCCCCCCC  RRRRRRRRRRRR        000000000
MMMMMM   MMMMMM  AAA         AAA  CCC              RRR         RRR  000         000
MMMMMM   MMMMMM  AAA         AAA  CCC              RRR         RRR  000         000
MMMMMM   MMMMMM  AAA         AAA  CCC              RRR         RRR  000         000
MMM   MMM   MMM  AAA         AAA  CCC              RRR         RRR  000         000
MMM   MMM   MMM  AAA         AAA  CCC              RRR         RRR  000         000
MMM   MMM   MMM  AAA         AAA  CCC              RRR         RRR  000         000
MMM         MMM  AAA         AAA  CCC              RRRRRRRRRRRR     000         000
MMM         MMM  AAA         AAA  CCC              RRRRRRRRRRRR     000         000
MMM         MMM  AAA         AAA  CCC              RRRRRRRRRRRR     000         000
MMM         MMM  AAAAAAAAAAAAAAA  CCC              RRR   RRR        000         000
MMM         MMM  AAAAAAAAAAAAAAA  CCC              RRR   RRR        000         000
MMM         MMM  AAAAAAAAAAAAAAA  CCC              RRR   RRR        000         000
MMM         MMM  AAA         AAA  CCC              RRR      RRR     000         000
MMM         MMM  AAA         AAA  CCC              RRR      RRR     000         000
MMM         MMM  AAA         AAA  CCC              RRR      RRR     000         000
MMM         MMM  AAA         AAA     CCCCCCCCCCCC  RRR         RRR     000000000
MMM         MMM  AAA         AAA     CCCCCCCCCCCC  RRR         RRR     000000000
MMM         MMM  AAA         AAA     CCCCCCCCCCCC  RRR         RRR     000000000
```

E 3

```
**FILE**ID**LINK

LL            IIIIII    NN      NN  KK      KK
LL            IIIIII    NN      NN  KK      KK
LL              II      NN      NN  KK      KK
LL              II      NN      NN  KK      KK
LL              II      NNNN    NN  KK    KK
LL              II      NNNN    NN  KK    KK
LL              II      NN  NN  NN  KKKKKK
LL              II      NN  NN  NN  KKKKKK
LL              II      NN    NNNN  KK    KK
LL              II      NN    NNNN  KK    KK
LL              II      NN      NN  KK      KK      ....
LL              II      NN      NN  KK      KK      ....
LLLLLLLLLL    IIIIII    NN      NN  KK      KK      ....
LLLLLLLLLL    IIIIII    NN      NN  KK      KK      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

MAC
V04

Table of contents

```
0000            .title  mac$link                link directive processor
0000            .ident  'V04-000'
0000    ;
0000    ;*****************************************************************************
0000    ;*                                                                           *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    ;*  ALL RIGHTS RESERVED.                                                     *
0000    ;*                                                                           *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    ;*  TRANSFERRED.                                                             *
0000    ;*                                                                           *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    ;*  CORPORATION.                                                             *
0000    ;*                                                                           *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    ;*                                                                           *
0000    ;*                                                                           *
0000    ;*****************************************************************************
0000    ;
0000    ;++
0000    ;
0000    ; Facility:
0000    ;
0000    ;       VAX-11 Macro Assembler
0000    ;
0000    ; Abstract:
0000    ;
0000    ;       This module contains the routines required to handle the .LINK
0000    ;       assembler directive.  The nature of this directive is to allow
0000    ;       the user to specify linker options within the object module
0000    ;       produced by the assembler.
0000    ;
0000    ; Environment:
0000    ;
0000    ;       Native Mode, User Mode
0000    ;
0000    ; Author:
0000    ;
0000    ;       Michael T. Rhodes,                      Creation Date: April, 1983
0000    ;
0000    ; Modified By:
0000    ;
0000    ;       V03-001 MTR0036         Michael T. Rhodes               16-Aug-1983
0000    ;               Add abbreviated qualifier name synonyms and adjust CASE
0000    ;               table dispatch address to accomodate the LNK$C_SHR object
0000    ;               record type.
0000    ;
0000    ;--
```

```
                   0000           .sbttl  declarations
                   0000
                   0000   ;
                   0000   ; Macros:
                   0000   ;
                   0000           $macmsgdef                      ; Define Macro-32's messages.
                   0000           $mac_ctlflgdef                  ; Define the control flags.
                   0000           $mac_objcoddef                  ; Define the object code.
                   0000           $mac_symblkdef                  ; Define the symbol block offsets etc..
                   0000
                   0000   ;
                   0000   ; Equated Symbols:
                   0000   ;
        0000000D   0000   cr = ^X0D                               ; Carriage return token.
        00000013   0000   lnk_blk_siz =   ^X13                    ; Size of the fixed portion of the linker op
        00000004   0000   lnk_q_inclst =  ^X4                     ; Offset to include list address.
        0000000C   0000   lnk_l_bytes =   ^XC                     ; Offset to #bytes required for include list
        00000010   0000   lnk_l_states =  ^X10                    ; Offset to link directive state flags.
                   0000
                   0000
            00000000      .psect  mac$rw_data, noexe, rd, wrt
                   0000   ;
                   0000   ; Linker option records list head.
                   0000   ;
                   0000   mac$gq_lnkopt::
        00000000'  0000           .address mac$gq_lnkopt
        00000000'  0004           .address mac$gq_lnkopt
                   0008
            00000000      .psect  mac$ro_data, noexe, nowrt, gbl, long
                   0000   ;
                   0000   ; Linker qualifier options table.
                   0000   ;
        00000000   0000   insymp = 0
                   0000           $mac_insert_syx SH,                     lnk$c_sha
                   000C           $mac_insert_syx SE,                     lnk$c_obj
                   0018           $mac_insert_syx I,                      lnk$c_oli
                   0023           $mac_insert_syx L,                      lnk$c_olb
                   002E           $mac_insert_syx SHAREABLE,              lnk$c_sha
                   0041           $mac_insert_syx SELECTIVE_SEARCH,       lnk$c_obj
                   005B           $mac_insert_syx INCLUDE,                lnk$c_oli
                   006C           $mac_insert_syx LIBRARY,                lnk$c_olb,      lnk_qualifiers
                   007D
```

```
MAC
Sym

BUI
CR
DSC
DSC
EOM
EOM
EOM
EOM
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
FLG
```

```
007D            .sbttl  .link  --  process the .link directive
007D    ;++
007D    ;
007D    ; Functional Description:
007D    ;
007D    ;       This routine is called to process the .LINK directive.
007D    ;       The valid syntax for this directive is as follows:
007D    ;
007D    ;               .LINK   "filespec"[/qualifier[=module or (module list)]],...
007D    ;
007D    ;       The filespec within the delimiters is scanned (built into a
007D    ;       .ASCID string), then we scan looking for a .LINK directive qualifier.
007D    ;       If none are present, the default linker option record type (regardless
007D    ;       of the extension specified in the filespec) is OBJECT (which includes
007D    ;       symbol tables).
007D    ;
007D    ; Implicit Inputs:
007D    ;
007D    ;       mac$ab_tmpbuf   adr     The address of the assembler's temporary buffer
007D    ;                               used to accumulate the delimited file specification.
007D    ;
007D    ;       mac$ab_tmpsym   adr     The address of the assembler's temporary buffer
007D    ;                               used to accumulate the qualifier name(s) and
007D    ;                               module name(s) specified in an include list.
007D    ;
007D    ;       mac$gq_lnkopt   adr     The address of the head of the linker options
007D    ;                               records queue.
007D    ;
007D    ; Implicit Outputs:
007D    ;
007D    ;       Linker option record(s) are placed into an ordered queue
007D    ;       (mac$gq_lnkopt) where the order is preserved FIFO.  This is done
007D    ;       to remain compatible with the LINKER's normal processing of option
007D    ;       records (as if they were speicified in a normal linker options file).
007D    ;       The list is subsequently written to the object file during pass 2,
007D    ;       following the object module header information.
007D    ;
007D    ; Special Case(s):
007D    ;
007D    ;       Special processing is performed when the /INCLUDE qualifier is specified.
007D    ;       The object module names contained in the include list are constructed as
007D    ;       .ASCIC strings following the filespec.  The record is terminated by a
007D    ;       module name string with a zero length.
007D    ;
007D    ;       If both the /LIBRARY and /INCLUDE qualifiers are specified for the
007D    ;       same library filespec, then the linker option record type is defaulted
007D    ;       to LNK$C_OLI and a special flag bit is set to indicate that the library
007D    ;       may be searched (LNK$V_LIBSRCH).
007D    ;
007D    ; Side Effects:
007D    ;
007D    ;       Two possible side effects can occur.  The first is a recoverable syntax
007D    ;       error where a diagnotic is issued to inform the user of the problem and
007D    ;       the assembly of the current input file continues.  The second is an
007D    ;       insufficient virtual memory error in which a diagnotic is issued to the
007D    ;       user and the assembly of the current input file is aborted.
007D    ;
```

```
                             007D   157 ; Stack Usage:
                             007D   158 ;
                             007D   159 ;        31           16      8       0
                             007D   160 ;        +------------+-------+-------+
                             007D   161 ;        !      flags ! lnktyp! rectyp!
                             007D   162 ;        +------------+-------+-------+
                             007D   163 ;        ! module name include list flink!
                             007D   164 ;        +----------------------------+
                             007D   165 ;        ! module name include list blink!
                             007D   166 ;        +----------------------------+
                             007D   167 ;        !   #bytes req for inclst     !
                             007D   168 ;        +----------------------------+
                             007D   169 ;        !   link directive states     !
                             007D   170 ;        +----------------------------+
                             007D   171 ;--
                             007D   172
                         00000000   173         .psect  mac$ro_code_p1, nowrt, gbl, long
                             0000   174
                             0000   175 link::                                          ; Directive = klink
           0100 8F      BB   0000   176         pushr   #^m<r8>                         ; Save registers.
   58   00000000'GF     9E   0004   177         movab   g^mac$ab_tmpbuf,r8              ; Address of temporary buffer.
    0000'C8    08 A8    9E   000B   178         movab   8(r8),dsc$a_pointer(r8)         ; Address of buffer to accumalate file name.
        5E     F0 AE    9E   0011   179         movab   -lnk_l_states(sp), sp           ; Allocate STACK LOCAL storage.
               10 AE    D4   0015   180         clrl    lnk_l_states(sp)                ; Initialize state flags.
                             0018   181
                             0018   182 ;
                             0018   183 ; Perform initialization...then get the [next] file name.
                             0018   184 ;
                  6E    D4   0018   185 10$:    clrl    (sp)                            ; Reset linker record type and flags.
               0C AE    D4   001A   186         clrl    lnk_l_bytes (sp)                ; Reset the include list byte count.
      04 AE    04 AE    9E   001D   187         movab   lnk_q_inclst (sp), lnk_q_inclst (sp)     ; Reset the queue list
      08 AE    04 AE    9E   0022   188         movab   lnk_q_inclst (sp), lnk_q_inclst+4 (sp)   ; head to empty condition.
         6E       06    90   0027   189         movb    #obj$c_lnk,lnk$b_rectyp(sp) ; Set the record type field.
                  58    DD   002A   190         pushl   r8                              ; Push address of descriptor/buffer.
   000000C3'EF    01    FB   002C   191         calls   #1, get_file_name               ; Get the [next] file name (if any).
            08    50    E8   0033   192         blbs    r0, 20$                         ; We got one...go check for qualifier(s).
      67 10 AE    00    E0   0036   193         bbs     #0,lnk_l_states(sp),70$ ; Have we processed at least one file?
            0073        31   003B   194         brw     90$                             ; No, unterminated macro argument error.
                             003E   195
                             003E   196 ;
                             003E   197 ; The file name has been accumulated, check for qualifiers.
                             003E   198 ;
                  68    B5   003E   199 20$:    tstw    (r8)                            ; Null file specification?
                  6F    13   0040   200         beql    80$                             ; Yes, directive syntax error.
      00000000'GF       16   0042   201         jsb     g^mac$getchr                    ; Skip over delimiter.
      00000000'GF       16   0048   202         jsb     g^mac$skipsp                    ; Skip spaces
            2F    5A    91   004E   203         cmpb    r10,#^A"/"                      ; Was the character a slash?
                  11    13   0051   204         beql    25$                             ; Yes, process the qualifier(s).
            2C    5A    91   0053   205         cmpb    r10,#^A","                      ; No, was it a comma (valid syntax)?
                  1A    13   0056   206         beql    30$                             ; Yes, use defaults.
            2D    5A    91   0058   207         cmpb    r10,#^A"-"                      ; Is the line continued?
                  15    13   005B   208         beql    30$                             ; Yes, finish current spec before continuing
            0D    5A    91   005D   209         cmpb    r10,#cr                         ; Have we reached the end of the input line?
                  10    13   0060   210         beql    30$                             ; Yes, normal terminator.
                  46    11   0062   211         brb     80$                             ; No, syntax error.
                  5E    DD   0064   212 25$:    pushl   sp                              ; Push the address of the link vector.
   00000106'EF    01    FB   0066   213         calls   #1, process_qual                ; Process the qualifier(s).
```

```
MAC
Pse

PSE
---
.
. B
$AB
MAC
MAC
MAC

Pha
---
Ini
Com
Pas
Sym
Pas
Sym
Pse
Cro
Ass

The
586
The
688
37

Mac
---
_$2
_$2
TOT

960

The

MAC
```

```
                 06 50   E8  006D          blbs    r0, 40$                  ; Use the option record type scanned.
                    38   11  0070          brb     80$                      ; Syntax error in qualifier(s).
                             0072
                             0072   ;
                             0072   ; Build the linker option record.
                             0072   ;
            01 AE   03   90  0072   30$:   movb    #lnk$c_obj,lnk$b_lnktyp(sp) ; Default option record type.
                    5E   DD  0076   40$:   pushl   sp                       ; Build a linker record using the
                    58   DD  0078          pushl   r8                       ; type and flags scanned above
      00000299'EF   02   FB  007A          calls   #2, build_lnk_rec        ; (Note: state flags are affected).
                             0081
                             0081   ;
                             0081   ; What's our next move...?
                             0081   ;
               2C   5A   91  0081          cmpb    r10, #^A","              ; Is the current character a comma?
                    08   13  0084          beql    50$                      ; Yes, go process the next list item.
               2D   5A   91  0086          cmpb    r10, #^A"-"              ; Is the line continued?
                    12   12  0089          bneq    60$                      ; No, check for eol.
               5A   0D   90  008B          movb    #cr, r10                 ; Yes, continue processing the directive.
      00000000'GF   16       008E   50$:   jsb     g^mac$getchr             ; Skip current character.
      00000000'GF   16       0094          jsb     g^mac$skipsp             ; Skip spaces, tabs, etc..
             FF7B   31       009A          brw     10$                      ; Continue
               0D   5A   91  009D   60$:   cmpb    r10,#cr                  ; Have we reached the end of the line?
                    08   12  00A0          bneq    80$                      ; No, report syntax error.
                             00A2
                             00A2   ;
                             00A2   ; We're done, clean up and return to the parser.
                             00A2   ;
               5E   10   C0  00A2   70$:   addl    #lnk_l_states, sp        ; Restore the stack.
             0100 8F     BA  00A5          popr    #^m<r8>                  ; Restore registers.
                         05  00A9          rsb                              ; Return to parser to continue.
                             00AA
                             00AA   ;
                             00AA   ; Some type of syntax error has been encountered...
                             00AA   ;
                             00AA   80$:   $mac_err dirsynx                 ; A directive syntax error has been
                    05   11  00AF          brb     100$                     ; encountered, issue error and return.
                             00B1
                             00B1   90$:   $mac_err untermarg               ; A terminator has not been seen for
                             00B6                                           ; the file name.
                             00B6
               5E   10   C0  00B6   100$:  addl    #lnk_l_states, sp        ; Restore the stack.
             0100 8F     BA  00B9          popr    #^m<r8>                  ; Restore current token.
      00000000'GF   17       00BD          jmp     g^mac$errorpt            ; Issue error message and return.
                             00C3
```

```
                              00C3          .sbttl  get_file_name   accumulate file name
                              00C3     ;++
                              00C3     ;
                              00C3     ; Functional Description:
                              00C3     ;
                              00C3     ;        This routine scans the input record for a delimitted file name.
                              00C3     ;
                              00C3     ; Inputs:
                              00C3     ;
                              00C3     ;        4(AP)   adr     The address of a descriptor which points to a buffer
                              00C3     ;                        to store the file name which we will scan.
                              00C3     ;
                              00C3     ; Outputs:
                              00C3     ;
                              00C3     ;        4(AP)   adr     The descriptor has been updated to reflect the size
                              00C3     ;                        of the file name which has been accumulated.
                              00C3     ;
                              00C3     ; Routine Value:
                              00C3     ;
                              00C3     ;        True    A file name has been scanned.
                              00C3     ;        False   No file name has been found (end of line or unterminated arg).
                              00C3     ;
                              00C3     ; Side Effects:
                              00C3     ;
                              00C3     ;        If there is no file name available (eg. we hit the end of the line)
                              00C3     ;        the length field of the descriptor will be zero upon exit or if the
                              00C3     ;        argument is unterminated, the length will be non-zero but the routine
                              00C3     ;        value will be false.
                              00C3     ;
                              00C3     ;--
                        0044  00C3          .entry  get_file_name ^m<r2,r6> ; Save registers upon entry.
                     51   DD  00C5          pushl   r1                      ; Preserve R1.
            51    04 AC   D0  00C7          movl    4(ap), r1               ; Get descriptor address.
                     61   D4  00CB          clrl    (r1)                    ; Initialize length, class, and type fields.
       52      0000'C1    D0  00CD          movl    dsc$a_pointer(r1), r2   ; Get the buffer address.
                     50   D4  00D2          clrl    r0                      ; Assume the worst...
        00000000'GF       16  00D4          jsb     g^mac$skipsp            ; Find the delimiter.
            0D       5A   91  00DA          cmpb    r10,#cr                 ; Have we reached the end of the line?
                     20   13  00DD          beql    30$                     ; Yes, return FALSE to the caller.
            56       5A   90  00DF          movb    r10,r6                  ; No, copy the delimiter and pass semi
            6B       01   C8  00E2          bisl2   #flg$m_allchr,(r11)     ; colons (to allow a version number).
                              00E5
        00000000'GF       16  00E5     10$: jsb     g^mac$getchr            ; Get the next character of the filename.
            56       5A   91  00EB          cmpb    r10,r6                  ; Is it the delimiter (end of filename)?
                     0C   13  00EE          beql    20$                     ; Yes, we're done here, return.
            0D       5A   91  00F0          cmpb    r10,#cr                 ; No, is it the end of the line?
                     0A   13  00F3          beql    30$                     ; Yes, upon return issue unterminated argume
            82       5A   90  00F5          movb    r10,(r2)+               ; No, store the character.
                     61   B6  00F8          incw    (r1)                    ; Keep track of file name length.
                     E9   11  00FA          brb     10$                     ; Gather the rest of the file name.
                              00FC
            50       01   D0  00FC     20$: movl    #1, r0                  ; Success...
                     51 8ED0  00FF     30$: popl    r1                      ; Restore R1.
            6B       01   CA  0102          bicl2   #flg$m_allchr,(r11)     ; Don't pass anymore semi-colons...
                          04  0105          ret
                              0106
                              0106              .sbttl  process_qual    process link directive qualifiers
                              0106       ;++
                              0106       ;
                              0106       ; Functional Description:
                              0106       ;
                              0106       ;       This routine processes the .LINK directive qualifiers.
                              0106       ;
                              0106       ; Inputs:
                              0106       ;
                              0106       ;       4(AP)   adr     Address of a linker record vector.
                              0106       ;
                              0106       ; Outputs:
                              0106       ;
                              0106       ;       4(AP)   adr     The linker record information is set in the vector.
                              0106       ;
                              0106       ; Routine Value:
                              0106       ;
                              0106       ;       True    Qualifiers have been processed without a problem.
                              0106       ;       False   There was a syntax error in either the qualifier name or in
                              0106       ;               the item list associated with the qualifier.
                              0106       ;
                              0106       ;--
                        01E0  0106              .entry  process_qual ^m<r5,r6,r7,r8>    ; Save registers.
          5E    F8 AE     9E  0108              movab   -8(sp), sp                      ; Get STACK LOCAL storage.
                   6E     D4  010C              clrl    (sp)                            ; Initialize done bit.
          57    04 AC     D0  010E              movl    4(ap), r7                       ; Get base adr of link info vector.
          58    04 A7     9E  0112              movab   lnk_q_inclst (r7), r8           ; Base address of file name list hea
                              0116
             2C    5A     91  0116       10$:   cmpb    r10, #^A/,/                     ; Did we stop on a comma?
                   0D     13  0119              beql    30$                             ; Yes, we're done with this file spe
             0D    5A     91  011B              cmpb    r10, #cr                        ; No, have we reached eol?
                   08     13  011E              beql    30$                             ; Yes, we're done, return.
             2F    5A     91  0120              cmpb    r10, #^A"/"                     ; Is the current character slash?
                   06     13  0123              beql    40$                             ; Yes, scan qualifier name.
                 00A9     31  0125       20$:   brw     110$                            ; No, syntax error.
                 00A1     31  0128       30$:   brw     100$                            ; Done, return success.
                              012B
          00000000'GF     16  012B       40$:   jsb     g^mac$getchr                    ; Yes, skip over it...
          00000000'GF     16  0131              jsb     g^mac$symscnup                  ; Get the qualifier name.
                EB 50     E9  0137              blbc    r0, 20$                         ; None found, error.
    55    00000074'EF     9E  013A              movab   lnk_qualifiers, r5              ; Use linker qualifier name table.
          00000000'GF     16  0141              jsb     g^mac$src_list                  ; Look up linker option qualifier.
                DB 50     E9  0147              blbc    r0, 20$                         ; Not found, error.
          00000000'GF     16  014A              jsb     g^mac$skipsp                    ; Position character pointer as need
                              0150
                              0150       ;
                              0150       ; Dispatch to appropriate processing routine.
                              0150       ;
    04    00    05 A1     CF  0150              casel   sym$l_val(r1), #0, #lnk$c_maxrectyp
                       000C'  0155       50$:   .word 60$-50$                           ; lnk$c_olb - /LIBRARY
                       002E'  0157              .word 70$-50$                           ; lnk$c_shr - (unsupported)
                       003A'  0159              .word 80$-50$                           ; lnk$c_oli - /INCLUDE=
                       0067'  015B              .word 90$-50$                           ; lnk$c_obj - /SELECTIVE_SEARCH
                       002E'  015D              .word 70$-50$                           ; lnk$c_sha - /SHAREABLE
                   64     11  015F              brb     95$                             ; Default, OBJ or STB
                              0161
```

```
                                0161  ;
                                0161  ; /LIBRARY                  Normal object library
                                0161  ;
           01 A7      95        0161  60$:    tstb    lnk$b_lnktyp (r7)                 ; Check for conflicting qualifiers.
              06      13        0164          beql    63$                               ; None specified.
      02   01 A7      91        0166          cmpb    lnk$b_lnktyp (r7), #lnk$c_oli     ; If /INCLUDE was specified, no conf
              65      12        016A          bneq    110$                              ; but anything else will conflict.
      01 A7   00      90        016C  63$:    movb    #lnk$c_olb, lnk$b_lnktyp (r7)     ; lnk$c_olb - Normal object library.
      00 6E   00      E2        0170          bbss    #0, (sp), .+1                     ; Indicate /LIBRARY has been specifi
      08 6E   01      E1        0174          bbc     #1, (sp), 65$                     ; If /INCLUDE has been specified, th
      02 A7   02      88        0178          bisb    #lnk$m_libsrch, lnk$w_flags (r7); the library should be searched and
      01 A7   02      90        017C          movb    #lnk$c_oli, lnk$b_lnktyp (r7)     ; type precedence goes to LNK$C_OLI.
            FF93      31        0180  65$:    brw     10$                               ; Get the next entity.
                                0183
                                0183  ;
                                0183  ; /SHAREABLE                Shareable Image
                                0183  ;
           01 A7      95        0183  70$:    tstb    lnk$b_lnktyp (r7)                 ; Check for conflicting qualifiers.
              49      12        0186          bneq    110$                              ; We have a conflict.
      01 A7   04      90        0188          movb    #lnk$c_sha, lnk$b_lnktyp (r7)     ; lnk$c_sha - Shareable Image
            FF87      31        018C          brw     10$                               ; Get the next entity.
                                018F
                                018F  ;
                                018F  ; /INCLUDE                  Object Library with Include list
                                018F  ;
           01 A7      95        018F  80$:    tstb    lnk$b_lnktyp (r7)                 ; Check for conflicting qualifiers.
              06      13        0192          beql    82$                               ; None specified.
      00   01 A7      91        0194          cmpb    lnk$b_lnktyp (r7), #lnk$c_olb     ; If /LIBRARY was specified, no conf
              37      12        0198          bneq    110$                              ; but anything else will conflict.
      01 A7   02      90        019A  82$:    movb    #lnk$c_oli, lnk$b_lnktyp (r7)     ; lnk$c_oli - Object Library with an
      00 6E   01      E2        019E          bbss    #1, (sp), .+1                     ; Indicate /INCLUDE has been specifi
      04 6E   00      E1        01A2          bbc     #0, (sp), 83$                     ; If /LIBRARY has been specified, th
      02 A7   02      88        01A6          bisb    #lnk$m_libsrch, lnk$w_flags (r7); the library should be searched.
           3D 5A      91        01AA  83$:    cmpb    r10, #^A/=/                       ; The next character should be an "=
              22      12        01AD          bneq    110$                              ; If not, its a syntax error.
   000001D4'EF 00     FB        01AF          calls   #0, get_incl_list                 ; Get the module name(s) in the incl
           18 50      E9        01B6          blbc    r0, 110$                          ; Issue syntax error.
            FF5A      31        01B9  85$:    brw     10$                               ; Cet the next entity.
                                01BC
                                01BC  ;
                                01BC  ; /SELECTIVE_SEARCH     Selective search of OLB or STB
                                01BC  ;
           01 A7      95        01BC  90$:    tstb    lnk$b_lnktyp (r7)                 ; Check for conflicting qualifiers.
              10      12        01BF          bneq    110$                              ; We have a conflict.
      02 A7   01      88        01C1          bisb    #lnk$m_selser, lnk$w_flags (r7)   ; lnk$v_selser - Selective search
      01 A7   03      90        01C5  95$:    movb    #lnk$c_obj, lnk$b_lnktyp (r7)     ; lnk$c_obj - Object Module
            FF4A      31        01C9          brw     10$                               ; Get the next entity.
                                01CC
                                01CC  ;
                                01CC  ; All done select status and return.
                                01CC  ;
         50   01      D0        01CC  100$:   movl    #1, r0                            ; Success.
              02      11        01CF          brb     120$                              ; Now return.
              50      D4        01D1  110$:   clrl    r0                                ; Error.
                      04        01D3  120$:   ret                                       ; Restore registers and return.
                                01D4
```

```
                               01D4        .sbttl  get_incl_list   Get the module(s) in the list
                               01D4  ;++
                               01D4  ;
                               01D4  ; Functional Description:
                               01D4  ;
                               01D4  ;       This routine scans the include list and produces a linked list
                               01D4  ;       containing the counted ascii strings of the module name(s) entered
                               01D4  ;       in the include list.
                               01D4  ;
                               01D4  ; Implicit Inputs:
                               01D4  ;
                               01D4  ;       r7              The address of the linker record.
                               01D4  ;       r8              The address of the module name include list head.
                               01D4  ;
                               01D4  ;       mac$ab_tmpsym   The address of a symbol just scanned.
                               01D4  ;
                               01D4  ; Outputs:
                               01D4  ;
                               01D4  ;       Module names obtained from the include list are added to the linked
                               01D4  ;       list.
                               01D4  ;
                               01D4  ;--
                         0000  01D4        .entry  get_incl_list ^m<>          ; Get the module names to include.
        5E    FC AE       9E   01D6        movab   -4(sp), sp                  ; Get LOCAL STORAGE.
                 6E       D4   01DA        clrl    (sp)                        ; Initialize local storage.
                               01DC
   00000000'GF            16   01DC  10$:  jsb     g^mac$getchr                ; Get the next character.
   00000000'GF            16   01E2        jsb     g^mac$skipsp                ; Skip spaces, tabs, etc..
        0D       5A       91   01E8        cmpb    r10, #cr                    ; End of line?
                 54       13   01EB        beql    80$                         ; Yes, syntax error.
                               01ED
        28       5A       91   01ED  20$:  cmpb    r10, #^A/(/                 ; Do we have a list of names?
                 25       13   01F0        beql    30$                         ; Yes, remove open paren and indicat
        2C       5A       91   01F2        cmpb    r10, #^A/,/                 ; Check for module name delimiter.
                 26       13   01F5        beql    40$                         ; Remove the comma, and validate str
        29       5A       91   01F7        cmpb    r10, #^A/)/                 ; Do we have a close paren?
                 2B       13   01FA        beql    50$                         ; Yes, end of the list?
        0D       5A       91   01FC        cmpb    r10, #cr                    ; End of line?
                 37       13   01FF        beql    60$                         ; We're done, select return status.
     00 6E       00       E4   0201        bbsc    #0, (sp), .+1               ; Reset comma seen flag.
   00000000'GF            16   0205        jsb     g^mac$symscnup              ; Get the module name.
           33    50       E9   020B        blbc    r0, 80$                     ; No file name, error.
0000024B'EF      00       FB   020E        calls   #0, insert_module           ; Insert this module name into the l
                 D6       11   0215        brb     20$                         ; Get the next module (if any).
                               0217
     26 6E       01       E2   0217  30$:  bbss    #1, (sp), 80$               ; Check for syntax error -- 2 or mor
                 BF       11   021B        brb     10$                         ; Indicate a list and continue the m
                               021D
     20 6E       01       E1   021D  40$:  bbc     #1, (sp), 80$               ; Comma seperated list not allowed o
     1C 6E       00       E2   0221        bbss    #0, (sp), 80$               ; To many commas?
                 B5       11   0225        brb     10$                         ; Remove comma, step to next module
                               0227
     16 6E       01       E1   0227  50$:  bbc     #1, (sp), 80$               ; Should we have a close paren?
   00000000'GF            16   022B        jsb     g^mac$getchr                ; Yes, skip it for correct grammatic
        68       58       D1   0231        cmpl    r8, (r8)                    ; Have we got at least one module?
                 0B       13   0234        beql    80$                         ; No, and we don't accept null lists
                 04       11   0236        brb     70$                         ; Everything looks ok, return succes
```

```
                      0238   488
 05 6E   01    E0     0238   489 60$:    bbs     #1, (sp), 80$                     ; Should we have a close paren? (pre
                      023C   490
    50   01    D0     023C   491 70$:    movl    #1, r0                            ; Parse successful...
         09    11     023F   492         brb     90$                               ; return success.
                      0241   493
         50    D4     0241   494 80$:    clrl    r0                                ; Parse failed.
    68   58    D0     0243   495         movl    r8, (r8)                          ; Fake an empty queue...
 04 A8   58    D0     0246   496         movl    r8, 4(r8)                         ; reset the list head.
               04     024A   497 90$:    ret                                       ; Return
                      024B   498
```

```
MAC
V04



FF

FF

01

02

FF

FF

FF

FF

FF

FF

FF

FF

FF

FF

FF

FF

00

FF

FF

FF

27

27

01

00

27

D8
```

```
                              024B        .sbttl  insert_module   Insert module name into list
                              024B  ;++
                              024B  ;
                              024B  ; Functional Descrtipion:
                              024B  ;
                              024B  ;       This routine will allocate a block of memory to store the
                              024B  ;       module name (counted asci string) and link it into the list
                              024B  ;       of other modules in the include list.
                              024B  ;
                              024B  ; Implicit Inputs:
                              024B  ;
                              024B  ;       r7      adr     Address of the linker options record control block.
                              024B  ;       r8      adr     Address of the head of the include list.
                              024B  ;
                              024B  ;       mac$ab_tmpsym   The buffer containing the counted asci string.
                              024B  ;
                              024B  ; Outputs:
                              024B  ;
                              024B  ;       The module has been added to the list.
                              024B  ;
                              024B  ; Side Effects:
                              024B  ;
                              024B  ;       If an error occurs while attempting to allocate dyanmic memory
                              024B  ;       we'll exit in the normal tradition (abort this assembly).
                              024B  ;
                              024B  ;--
                        027C  024B        .entry  insert_module ^m<r2,r3,r4,r5,r6,r9>
                     51   DD  024D        pushl   r1                      ; Preserve R1.
     56   00000000'GF     9E  024F        movab   g^mac$ab_tmpsym, r6     ; Get the beginning address of the buffer.
                 59  66   9A  0256        movzbl  (r6), r9                ; Get the size of the string.
                     59   D6  0259        incl    r9                      ; Bump the string count to include count byt
                 5E  08   C2  025B        subl2   #8, sp                  ; Get STACK LOCAL storage.
                 04  AE   D4  025E        clrl    4(sp)                   ; Initialize the return address scalar.
                 04  AE   9F  0261        pushab  4(sp)                   ; Push address of the return address scalar.
         04 AE   59  08   C1  0264        addl3   #8, r9, 4(sp)           ; Compute the number of bytes to allocate.
                 04  AE   9F  0269        pushab  4(sp)                   ; Push the address of the number of bytes.
       00000000'GF   02   FB  026C        calls   #2, g^lib$get_vm        ; Allocate the module name block.
                 16  50   E9  0273        blbc    r0, ins_vir_mem         ; Insufficient Vitrual Memory error.
             53  04  AE   D0  0276        movl    4(sp), r3               ; Get the beginning address of the module na
         08 A3   66  59   28  027A        movc3   r9, (r6), 8(r3)         ; Copy the string.
          04 B8  04  BE   0E  027F        insque  @4(sp), @4(r8)          ; Insert this block at the tail of the list.
             0C  A7  59   C0  0284        addl2   r9, lnk_l_bytes (r7)    ; Add this string's byte count to the sum.
                     51 8ED0  0288        popl    r1                      ; Restore R1.
                          04  028B        ret
                              028C
                              028C  ;
                              028C  ; Insufficient Virtual Memory, report error and abort this assembly.
                              028C  ;
                              028C  ins_vir_mem:
       00000000'GF   00   FB  028C        calls   #0, g^mac$err_nomem_0   ; Report error.
          00000000'GF     17  0293        jmp     g^mac$last_chance       ; Abort this assembly.
                              0299
```

```
0299          .sbttl  build_lnk_rec   Build a linker options record
0299    ;++
0299    ;
0299    ; Functional Description:
0299    ;
0299    ;       This routine builds a linker options record.  First we obtain
0299    ;       a linker options record block (by a call to LIB$GET_VM), next
0299    ;       fill in the information and link it into the queue of linker
0299    ;       options records and set the 'at least one file processed' bit
0299    ;       in the state flags.  Special handling of the object library with
0299    ;       an include list is performed by copying the strings from the
0299    ;       include module list and deallocating the module name blocks.
0299    ;
0299    ;       31              16       8       0
0299    ;       +---------------+--------+-------+
0299    ;       !             flink              !
0299    ;       +--------------------------------+
0299    ;       !             blink              !
0299    ;       +--------------------------------+
0299    ;       !size of this lnk opt rec block*!        * NOTE: The size field contains
0299    ;       +---------------+--------+-------+               a value which represents
0299    ;       ! flags         ! lnktyp! rectyp!                the total size of the
0299    ;       +---------------+--------+-------+               block in bytes.
0299    ;       !               !      namlng    !               It includes the overhead
0299    ;       !               +----------------+               information (flink, blink,
0299    ;       !       file specification       !               size field and the terminato
0299    ;       !                                !
0299    ;       +--------------------------------+
0299    ;       !                        ! count !
0299    ;       ! optional module        +-------+
0299    ;       ! name from /INCLUDE=[()] list   !
0299    ;       +--------------------------------+
0299    ;       !               :                !
0299    ;       !               :                !
0299    ;       +--------------------------------+
0299    ;       !                        ! count !
0299    ;       ! optional module        +-------+
0299    ;       ! name from /INCLUDE=[()] list   !
0299    ;       +--------------------------------+
0299    ;                                !   0   ! Terminator byte (module name length 0)
0299    ;                                +-------+
0299    ; Inputs:
0299    ;
0299    ;       4(AP)           adr     Address of the filespec descriptor.
0299    ;       8(AP)           adr     Address of the linker record control block.
0299    ;
0299    ; Implicit Inputs:
0299    ;
0299    ;       mac$gq_lnkopt   adr     The address of the linker options record queue
0299    ;                               list head.
0299    ;
0299    ; Side Effects:
0299    ;
0299    ;       If an error occurs while attempting to allocate dyanmic memory
0299    ;       we'll exit in the normal tradition (abort this assembly).
0299    ;
0299    ;--
```

```
                       01FC 0299          .entry  build_lnk_rec ^m<r2,r3,r4,r5,r6,r7,r8>
          52    04 AC    D0 029B          movl    4(ap), r2               ; Get the address of the filespec.
          57    08 AC    D0 029F          movl    8(ap), r7               ; Get link control block address.
             5E    08    C2 02A3          subl2   #8, sp                  ; Allocate STACK LOCAL storage.
                04 AE    D4 02A6          clrl    4(sp)                   ; Initialize the return address scalar.
                04 AE    9F 02A9          pushab  4(sp)                   ; Push the address of the return address sca
       04 AE    0000'C2  3C 02AC          movzwl  dsc$w_length (r2),4(sp) ; Get the length.
04 AE    0C A7    04 AE  C1 02B2          addl3   4(sp),lnk_l_bytes(r7),4(sp) ; Compute the #bytes req for this record
          04 AE    13    C0 02B9          addl2   #lnk_blk_siz, 4(sp)     ; Include the fixed area size in the count.
                04 AE    9F 02BD          pushab  4(sp)                   ; Address of the number of byte req.
      00000000'GF  02    FB 02C0          calls   #2, g^lib$get_vm        ; Allocate memory for this linker options re
             C2    50    E9 02C7          blbc    r0, ins_vir_mem         ; Insufficient Virtual Memory error.
          53    04 AE    D0 02CA          movl    4(sp), r3               ; Get the beginning address of the linker op
             58    53    D0 02CE          movl    r3, r8                  ; Preserve the block address for later use.
             53    08    C0 02D1          addl2   #8, r3                  ; Advance pointer to first data field.
             83    6E    D0 02D4          movl    (sp), (r3)+             ; Copy the number of bytes in this block.
             83    67    D0 02D7          movl    (r7), (r3)+             ; Copy the record type and flags word to the
             83    62    B0 02DA          movw    (r2), (r3)+             ; Copy the file spec length.
    63    0000'D2  62    28 02DD          movc3   (r2),@dsc$a_pointer(r2),(r3) ; Copy the file spec.
                            02E3
          56    04 B7    0F 02E3  10$:    remque  @lnk_q_inclst(r7), r6   ; Remove the next module name.
             26    1D       02E7          bvs     20$                     ; Is the queue empty?
          04 AE    56    D0 02E9          movl    r6, 4(sp)               ; No, get the address of the module name
                04 AE    9F 02ED          pushab  4(sp)                   ; block to release and pass it by reference.
       04 AE    08 A6    9A 02F0          movzbl  8(r6), 4(sp)            ; Get the string length and include
          04 AE    01    80 02F5          addb2   #1, 4(sp)               ; the count byte in the string size.
    63    08 A6    04 AE 28 02F9          movc3   4(sp), 8(r6), (r3)      ; Copy the module size/name to the record.
          04 AE    08    C0 02FF          addl2   #8, 4(sp)               ; The linkage is included in the mnb size.
                04 AE    9F 0303          pushab  4(sp)                   ; Pass the block size by reference too.
      00000000'GF  02    FB 0306          calls   #2, g^lib$free_vm       ; Release the module name block.
                   D4    11 030D          brb     10$                     ; Get the next module.
                            030F
                   63    94 030F  20$:    clrb    (r3)                    ; Mark include list terminator.
    56    00000000'GF    9E 0311          movab   g^mac$gq_lnkopt, r6     ; Get the address of the linker options queu
          04 B6    68    0E 0318          insque  (r8), @4(r6)            ; Insert this record into the linker options
    00 10 A7       00    E2 031C          bbss    #0,lnk_l_states(r7),.+1 ; Set flag indicating at least 1 file proces
                         04 0321          ret
                            0322
```

```
MAC
V04

00
01
00
00
02
00
01
00
00
01
01
00
02
02
02
00
00
00
01
04
```

```
                               0322          .sbttl  mac$wrt_lnkopt  Write the linker options records to object
                               0322     ;++
                               0322     ;
                               0322     ; Functional Description:
                               0322     ;
                               0322     ;        This routine removes the linker options records from the queue
                               0322     ;        MAC$GQ_LNKOPT and writes them to the object module (following the GSD).
                               0322     ;
                               0322     ; Implicit Inputs:
                               0322     ;
                               0322     ;        R10             adr     Contains the address of the object code buffer.
                               0322     ;        mac$gq_lnkopt   adr     The address of the linker option record queue.
                               0322     ;
                               0322     ; Side Effects:
                               0322     ;
                               0322     ;        All linker option record(s) have been written to the object file
                               0322     ;        and the currect object record buffer type will be seL to OBJ$C_TIR
                               0322     ;        upon exit.
                               0322     ;
                               0322     ;--
                         0040  0322          .entry  mac$wrt_lnkopt ^m<r6>    ; Save register(s).
           5E     08       C2  0324          subl2   #8, sp                   ; Allocate STACK LOCAL storage.
  56    00000000'FF       0F   0327 10$:     remque  @mac$gq_lnkopt, r6       ; Get a linker option record.
                  2A       1D  032E          bvs     20$                      ; Is the queue empty?
                  5A       D7  0330          decl    r10                      ; No, set the buffer pointer to origin.
  51     08 A6    0C       C3  0332          subl3   #12, 8(r6), r1           ; Compute the size of this record.
  6A     0C A6    51       28  0337          movc3   r1, 12(r6), (r10)        ; Copy the record to the object code buffer.
           5A     53       D0  033C          movl    r3, r10                  ; Update the object code pointer.
              FCBE'        30  033F          bsbw    mac$wrtobj               ; Write the object record.
        04 AE     56       D0  0342          movl    r6, 4(sp)                ; Release dynamic memory...
              04 AE        9F  0346          pushab  4(sp)                    ; Pass the block's address by reference.
    04 AE    08 A6         D0  0349          movl    8(r6), 4(sp)             ; The linker option record's block size
              04 AE        9F  034E          pushab  4(sp)                    ; is also passed by reference.
  00000000'GF     02       FB  0351          calls   #2, g^lib$free_vm        ; Release this block.
                  CD       11  0358          brb     10$                      ; Continue until the queue is empty.
                               035A
           6A     02       90  035A 20$:     movb    #obj$c_tir, (r10)        ; All done, correct the object record
                           04  035D          ret                              ; type to assume TIR.
                               035E
                               035E          .end    ; of MODULE mac$link
```

```
                                                 H  4
MAC$LINK                     link directive processor                16-SEP-1984 02:06:27  VAX/VMS Macro V04-00          Page  15
Symbol table                                                          5-SEP-1984 01:48:43  [MACRO.SRC]LINK.MAR;1               (11)

BUILD_LNK_REC    00000299 RG     05   FLG$M_UPMARG   = 00000040             INSYMP         = 00000074 R      04
CR             = 0000000D             FLG$M_XCRF     = 80000000             INSYTM         = 00000074 R      04
DSC$A_POINTER    ********      X  05  FLG$V_ALLCHR   = 00000000             INS_VIR_MEM      0000028C R      05
DSC$W_LENGTH     ********      X  05  FLG$V_BOL      = 00000001             LIB$FREE_VM      ********      X  05
EOM$C_ABORT    = 00000003             FLG$V_CHKLPND  = 00000014             LIB$GET_VM       ********      X  05
EOM$C_ERROR    = 00000002             FLG$V_COMPEXPR = 00000002             LINK             00000000 RG     05
EOM$C_SUCCESS  = 00000000             FLG$V_CONT     = 00000003             LNK$B_LNKTYP   = 00000001
EOM$C_WARNING  = 00000001             FLG$V_CRF      = 0000001E             LNK$B_RECTYP   = 00000000
FLG$M_ALLCHR   = 00000001             FLG$V_CRSEEN   = 00000020             LNK$C_MAXRECTYP= 00000004
FLG$M_BOL      = 00000002             FLG$V_DATRPT   = 00000004             LNK$C_OBJ      = 00000003
FLG$M_CHKLPND  = 00100000             FLG$V_DBGOUT   = 0000002E             LNK$C_OLB      = 00000000
FLG$M_COMPEXPR = 00000004             FLG$V_DLIMSTR  = 0000002F             LNK$C_OLI      = 00000002
FLG$M_CONT     = 00000008             FLG$V_ENDMCH   = 00000005             LNK$C_SHA      = 00000004
FLG$M_CRF      = 40000000             FLG$V_EVALEXPR = 00000006             LNK$M_LIBSRCH  = 00000002
FLG$M_CRSEEN   = 00000001             FLG$V_EXPOPT   = 00000007             LNK$M_SELSER   = 00000001
FLG$M_DATRPT   = 00000010             FLG$V_EXTERR   = 00000030             LNK$W_FLAGS    = 00000002
FLG$M_DBGOUT   = 00004000             FLG$V_EXTWRN   = 00000031             LNK_BLK_SIZ    = 00000013
FLG$M_DLIMSTR  = 00008000             FLG$V_FIRSTLN  = 00000029             LNK_L_BYTES    = 0000000C
FLG$M_ENDMCH   = 00000020             FLG$V_IFSTAT   = 00000017             LNK_L_STATES   = 00000010
FLG$M_EVALEXPR = 00000040             FLG$V_IIF      = 00000016             LNK_QUALIFIERS   00000074 RG     04
FLG$M_EXPOPT   = 00000080             FLG$V_INSERT   = 00000008             LNK_Q_INCLST   = 00000004
FLG$M_EXTERR   = 00010000             FLG$V_IRPC     = 0000001D             MAC$AB_TMPBUF    ********      X  05
FLG$M_EXTWRN   = 00020000             FLG$V_LEXOP    = 00000021             MAC$AB_TMPSYM    ********      X  05
FLG$M_FIRSTLN  = 00000200             FLG$V_LSTXST   = 00000009             MAC$ERRORPT      ********      X  05
FLG$M_IFSTAT   = 00800000             FLG$V_MAC2COL  = 0000002B             MAC$ERR_NOMEM_O  ********      X  05
FLG$M_IIF      = 00400000             FLG$V_MACL     = 0000000B             MAC$GETCHR       ********      X  05
FLG$M_INSERT   = 00000100             FLG$V_MACLTB   = 0000001B             MAC$GQ_LNKOPT    00000000 RG     03
FLG$M_IRPC     = 20000000             FLG$V_MACTXT   = 00000010             MAC$LAST_CHANCE  ********      X  05
FLG$M_LEXOP    = 00000002             FLG$V_MEBLST   = 0000000C             MAC$SKIPSP       ********      X  05
FLG$M_LSTXST   = 00000200             FLG$V_MOREARG  = 0000002D             MAC$SRC_LIST     ********      X  05
FLG$M_MAC2COL  = 00000800             FLG$V_MOREINP  = 00000023             MAC$SYMSCNUP     ********      X  05
FLG$M_MACL     = 00000800             FLG$V_NEWPND   = 0000000A             MAC$WRTOBJ       ********      X  05
FLG$M_MACLTB   = 08000000             FLG$V_NOREF    = 00000018             MAC$WRT_LNKOPT   00000322 RG     05
FLG$M_MACTXT   = 00010000             FLG$V_NTYPEPC  = 00000025             MAC$_DIRSYNX   = 007D906A
FLG$M_MEBLST   = 00001000             FLG$V_NULCHR   = 00000032             MAC$_UNTERMARG = 007D922A
FLG$M_MOREARG  = 00002000             FLG$V_OBJXST   = 00000015             OBJ$C_EOM_ABORT= 00000003
FLG$M_MOREINP  = 00000008             FLG$V_OPNDCHK  = 00000028             OBJ$C_EOM_ERR  = 00000002
FLG$M_NEWPND   = 00000400             FLG$V_OPRND    = 0000000D             OBJ$C_EOM_OK   = 00000000
FLG$M_NOREF    = 01000000             FLG$V_OPTVFLIDX= 0000002C             OBJ$C_EOM_WRN  = 00000001
FLG$M_NTYPEPC  = 00000020             FLG$V_ORDLST   = 00000011             OBJ$C_LNK      = 00000006
FLG$M_NULCHR   = 00040000             FLG$V_P2       = 0000000E             OBJ$C_TIR      = 00000002
FLG$M_OBJXST   = 00200000             FLG$V_RPTIRP   = 0000001C             OPF$M_LASTOPR  = 00002000
FLG$M_OPNDCHK  = 00000100             FLG$V_SEQFIL   = 00000019             OPF$M_OPTEXP   = 00001000
FLG$M_OPRND    = 00002000             FLG$V_SKAN     = 0000000F             OPF$V_LASTOPR  = 0000000D
FLG$M_OPTVFLIDX= 00001000             FLG$V_SPECOP   = 00000022             OPF$V_OPTEXP   = 0000000C
FLG$M_ORDLST   = 00020000             FLG$V_SPLALL   = 0000001A             PROCESS_QUAL     00000106 RG     05
FLG$M_P2       = 00004000             FLG$V_STOIMF   = 00000012             PSC$B_NAME       00000004
FLG$M_RPTIRP   = 10000000             FLG$V_SYM2COL  = 0000002A             PSC$B_SEG        0000000C
FLG$M_SEQFIL   = 02000000             FLG$V_TOCFLG   = 00000013             PSC$B_UNUSED     0000000B
FLG$M_SKAN     = 00008000             FLG$V_UPAFLG   = 00000024             PSC$K_BLKSIZ     00000013
FLG$M_SPECOP   = 00000004             FLG$V_UPDFIL   = 00000027             PSC$K_NO_OPTNS = 0000000A
FLG$M_SPLALL   = 04000000             FLG$V_UPMARG   = 00000026             PSC$L_CURLOC     0000000F
FLG$M_STOIMF   = 00040000             FLG$V_XCRF     = 0000001F             PSC$L_LINK       00000000
FLG$M_SYM2COL  = 00000400             GET_FILE_NAME    000000C3 RG     05   PSC$L_MAXLGTH    00000005
FLG$M_TOCFLG   = 00080000             GET_INCL_LIST    000001D4 RG     05   PSC$M_ABS      = FFFFFFF7
FLG$M_UPAFLG   = 00000010             INSERT_MODULE    0000024B RG     05   PSC$M_ALIGNFLG = 00004000
FLG$M_UPDFIL   = 00000080             INSYMC         = 00000007             PSC$M_ALLOPTNS = 000003FF
```

```
PSC$M_BYTE       = 00004000                SYM$M_DELMAC     = 00000200
PSC$M_CON        = FFFFFFFB                SYM$M_EPT        = 00000200
PSC$M_DEFAULT    = 000001C8                SYM$M_EXTRN      = 00000008
PSC$M_EXE        = 000000C0                SYM$M_GLOBL      = 00000004
PSC$M_GBL        = 00000010                SYM$M_LOCAL      = 00000040
PSC$M_LCL        = FFFFFFEF                SYM$M_ODBG       = 00000400
PSC$M_LIB        = 00000002                SYM$M_REF        = 00000080
PSC$M_LONG       = 00004800                SYM$M_RELPSECT   = 00000800
PSC$M_NOEXE      = FFFFFFBF                SYM$M_SUPR       = 00004000
PSC$M_NOPIC      = FFFFFFFE                SYM$M_WEAK       = 00000002
PSC$M_NORD       = FFFFFF7F                SYM$M_XCRF       = 00001000
PSC$M_NOSHR      = FFFFFFDF                SYM$V_ABS        = 00000004
PSC$M_NOVEC      = FFFFFDFF                SYM$V_ASN        = 00000008
PSC$M_NOWRT      = FFFFFEFF                SYM$V_CRFO       = 0000000D
PSC$M_OVR        = 00000004                SYM$V_DEBUG      = 00000005
PSC$M_PAGE       = 00006400                SYM$V_DEF        = 00000000
PSC$M_PIC        = 00000001                SYM$V_DELMAC     = 00000009
PSC$M_QUAD       = 00004C00                SYM$V_EPT        = 00000009
PSC$M_RD         = 00000080                SYM$V_EXTRN      = 00000003
PSC$M_REL        = 00000008                SYM$V_GLOBL      = 00000002
PSC$M_SHR        = 00000020                SYM$V_LOCAL      = 00000006
PSC$M_USR        = FFFFFFFD                SYM$V_ODBG       = 0000000A
PSC$M_VEC        = 00000200                SYM$V_REF        = 00000007
PSC$M_WORD       = 00004400                SYM$V_RELPSECT   = 0000000B
PSC$M_WRT        = 00000180                SYM$V_SUPR       = 0000000E
PSC$S_ALIGNMENT= 00000004                  SYM$V_WEAK       = 00000001
PSC$V_ALIGNFLG = 0000000E                  SYM$V_XCRF       = 0000000C
PSC$V_ALIGNMENT= 0000000A                  SYM$W_FLAG         00000009
PSC$V_EXE        = 00000006                TIR$C_STO_L      = 00000016
PSC$V_GBL        = 00000004                TIR$C_STO_LW     = 00000016
PSC$V_LIB        = 00000001                X1               = 00000400
PSC$V_OVR        = 00000002                X2               = 0000000F
PSC$V_PIC        = 00000000
PSC$V_RD         = 00000007
PSC$V_REL        = 00000003
PSC$V_SHR        = 00000005
PSC$V_VEC        = 00000009
PSC$V_WRT        = 00000008
PSC$W_FLAG         00000009
PSC$W_OPTIONS      0000000D
SYM$B_NAME         00000004
SYM$B_SEG          0000000C
SYM$B_TOKEN        0000000B
SYM$F_DEF        = 00000002
SYM$F_REL        = 00000008
SYM$F_UNI        = 00000004
SYM$F_VALIDATE   = 00000010
SYM$F_WEAK       = 00000001
SYM$K_BLKSIZ       0000000D
SYM$K_MAXLEN     = 0000001F
SYM$L_LINK         00000000
SYM$L_VAL          00000005
SYM$M_ABS        = 00000010
SYM$M_ASN        = 00000100
SYM$M_CRFO       = 00002000
SYM$M_DEBUG      = 00000020
SYM$M_DEF        = 00000001
```

```
MAC$LINK                          link directive processor          J  4       16-SEP-1984 02:06:27  VAX/VMS Macro V04-00         Page  17
Psect synopsis                                                                  5-SEP-1984 01:48:43  [MACRO.SRC]LINK.MAR;1              (11)

                                                       +----------------+
                                                       ! Psect synopsis !
                                                       +----------------+

PSECT name                        Allocation          PSECT No.  Attributes
----------                        ----------          ---------  ----------
.  ABS  .                         00000000  (     0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
. BLANK .                         00000000  (     0.)  01 (  1.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$ABS$                             00000013  (    19.)  02 (  2.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
MAC$RW_DATA                       00000008  (     8.)  03 (  3.)  NOPIC    USR    CON    REL    LCL NOSHR NOEXE   RD     WRT NOVEC BYTE
MAC$RO_DATA                       0000007D  (   125.)  04 (  4.)  NOPIC    USR    CON    REL    GBL NOSHR NOEXE   RD   NOWRT NOVEC LONG
MAC$RO_CODE_P1                    0000035E  (   862.)  05 (  5.)  NOPIC    USR    CON    REL    GBL NOSHR   EXE   RD   NOWRT NOVEC LONG

                                                 +------------------------+
                                                 ! Performance indicators !
                                                 +------------------------+

Phase                        Page faults     CPU Time       Elapsed Time
-----                        -----------     --------       ------------
Initialization                        32     00:00:00.06    00:00:00.46
Command processing                   108     00:00:00.38    00:00:03.67
Pass 1                               340     00:00:07.38    00:00:31.50
Symbol table sort                      0     00:00:00.82    00:00:02.10
Pass 2                               137     00:00:01.81    00:00:03.82
Symbol table output                   22     00:00:00.12    00:00:00.12
Psect synopsis output                  3     00:00:00.03    00:00:00.03
Cross-reference output                 0     00:00:00.00    00:00:00.00
Assembler run totals                 644     00:00:10.60    00:00:41.70

The working set limit was 1650 pages.
58640 bytes (115 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 787 non-local and 45 local symbols.
688 source lines were read in Pass 1, producing 36 object records in Pass 2.
37 pages of virtual memory were used to define 36 macros.

                                                 +----------------------------+
                                                 ! Macro library statistics !
                                                 +----------------------------+

Macro library name                               Macros defined
------------------                               --------------
_$255$DUA28:[MACRO.OBJ]MACRO.MLB;1                            6
_$255$DUA28:[SYSLIB]STARLET.MLB;2                             4
TOTALS (all libraries)                                       10

960 GETS were required to define 10 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:LINK/OBJ=OBJ$:LINK MSRC$:LINK/UPDATE=(ENH$:LINK)+LIB$:MACRO/LIB
```

```
MAC
Sym

ADD
AIN
AIN
AIN
ALI
ALI
ALI
ASC
ASC
ASC
ASC
ASS
ASS
ASS
ASS
AUT
AUT
AUT
AUT
BDE
BDE
BLK
BLK
BLK
BLK
BLK
BST
BST
BYT
CHA
CHR
CHR
CRE
CRO
CRO
CSA
DAL
DAL
DAT
DAT
DEB
DEL
DFN
DIR
DIS
DIS
DIS
DIS
DIS
DIS
DIS
DIS
DIS
DIS
DIS
DOU
DSA
ENA
```

0226 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY